

# Generación dinámica en el lado del servidor.

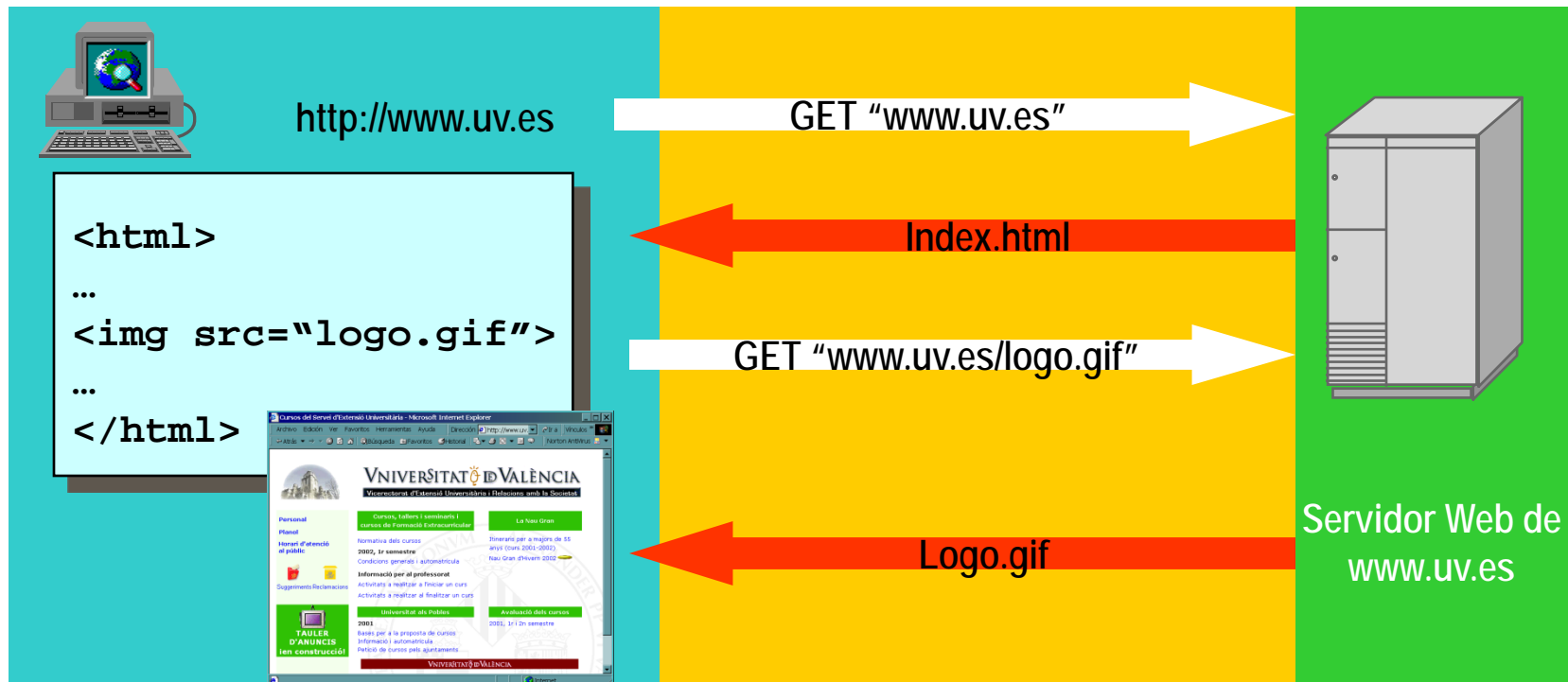
---

1. Introducción.
2. CGIs.
3. PHP: Marco.
4. PHP: Elementos.

## 1. Introducción

# Páginas estáticas

- El XHTML (y las imágenes que incluye) es algo **estático**.
  - Todo acceso a un documento, da el mismo resultado.
  - No satisface las necesidades de interactividad.



# Páginas dinámicas

---

- Más interactividad con el usuario (formularios) y dinamismo.
- **Generación dinámica en el lado del servidor:** CGI, ASP, PHP, JSP, etc.
- **En el lado del Cliente**
  - **Lenguaje de script** en el cliente. Se integran dentro del código. Javascript y VBscript.
  - **Hojas de estilo (CSS):** Definen plantillas, características no exactamente dinámica. Combinadas con javascript (y su modelo de objetos) crean páginas dinámicas.
  - **DHTML:** páginas con contenido multimedia más rico.
    - DHTML = CSS + javascript + Modelo de objetos del documento (DOM) + Flash.
    - Problema: no es estándar.
  - **Applets.**

# Generación dinámica en el servidor (I)

---

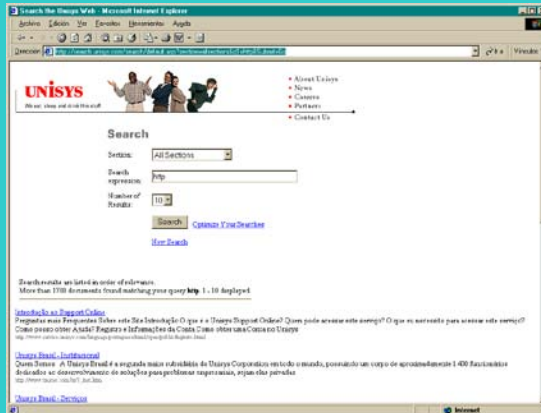
- Limitaciones de los documentos XHTML estáticos:
  - No adaptan los documentos a clientes individuales.
  - Hay problemas para actualizar los documentos, especialmente si parte de los datos están replicados en varios documentos.
  - Son imposibles las aplicaciones web (comercio electrónico).
- Solución: Generar los documentos Web dinámicamente en el lado del servidor, en el momento de la petición.

## 1. Introducción

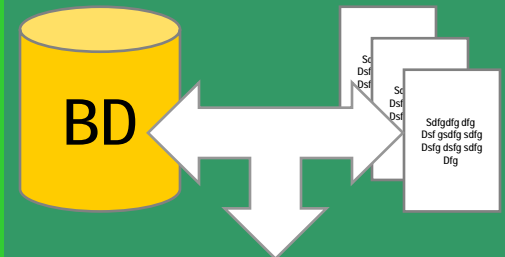
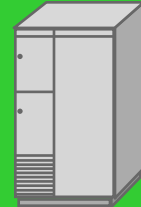
# Generación dinámica en el servidor (II)



<http://search.unisys.com/search/default.asp?section=allsections&q1=http&Submit=Go>



GET "...&Submit=Go"



Respuesta dinámica

Servidor web de  
[www.unisys.com](http://www.unisys.com)

# Ejemplos de usos de la generación dinámica

---

- **Cualquier aplicación web de comercio electrónico.**
- Recoger y procesar la información que proviene de un formulario.
  - Generar dinámicamente contenido a partir de esa información
- Generar dinámicamente información a partir de parámetros cambiantes con el tiempo.
  - p.e. un periódico electrónico, información bursátil, etc.
- Generar dinámicamente contenido a partir de la consulta a una base de datos.
  - p.e. los motores de búsqueda.
- Contar el número de accesos a la página.
- Anuncios incrustados (*banners*).
- Etc.

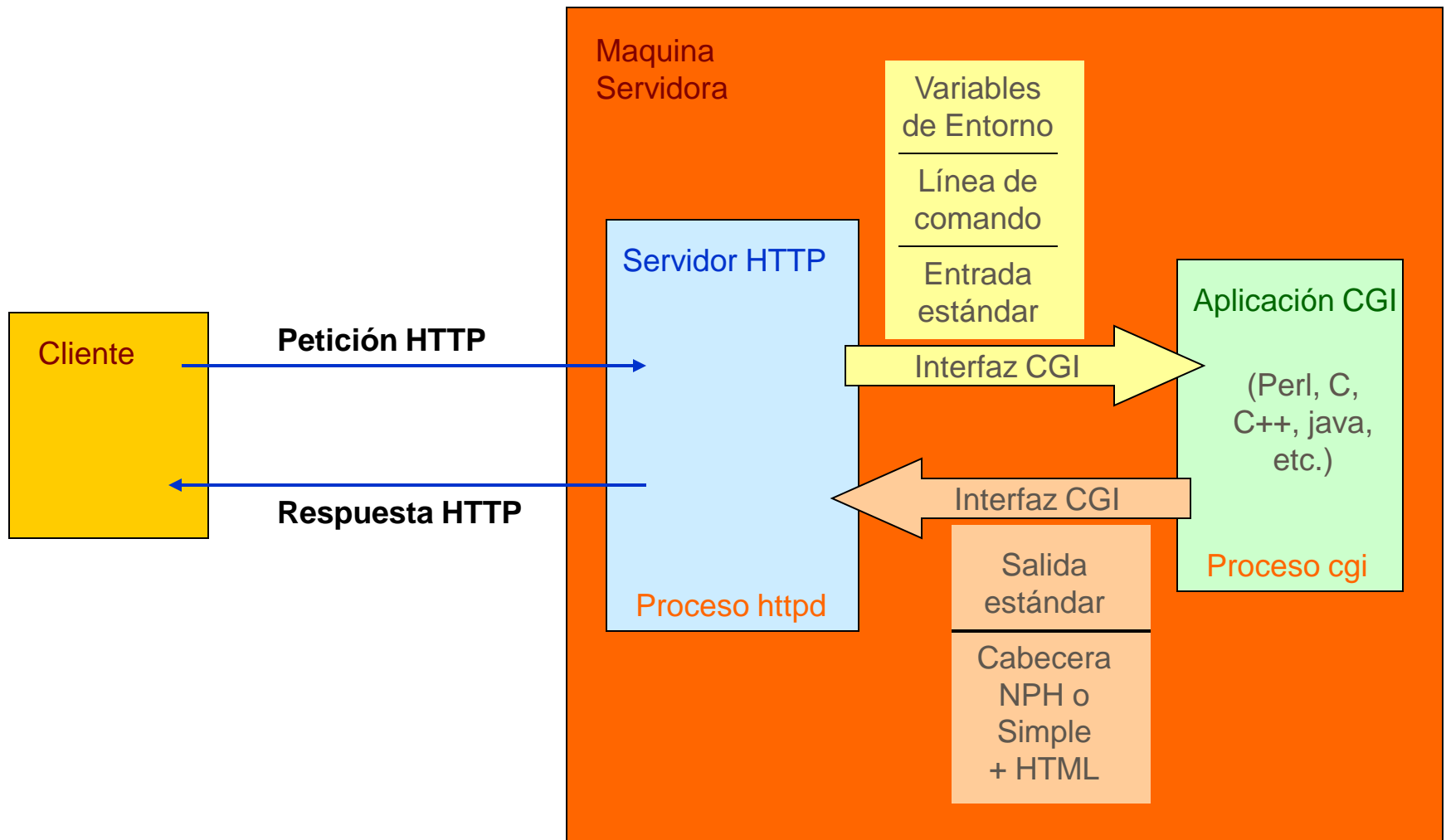
# Introducción a los CGIs

---

- NCSA creó el **Common Gateway Interface (CGI)**.
  - Rige como interaccionan los servidores HTTP con programas especiales que se ejecutan en la máquina servidora, destinados a procesar la información que envían los clientes Web y generar documentos de forma dinámica.
- A estos programas especiales se les llama **aplicaciones o módulos CGI** (la mayoría de veces, simplemente **CGI** ).
  - Son la primera de una serie de soluciones ideadas para publicar información dinámica (extraída normalmente de una BD).
    - El contenido es cambiante con el tiempo, dependiendo del entorno, del usuario, de los datos enviados, etc.

## 2. CGI

# Funcionamiento básico (I)





# Funcionamiento básico (II)

---

- El usuario:
  - Rellena un formulario HTML y envía los datos.
  - La acción va asociada a la URL de una aplicación CGI.
- El servidor Web:
  - Lanza la aplicación CGI como un proceso independiente.
  - Pasa los datos del formulario a través de: variables de entorno, línea de comandos y entrada estándar.
- La aplicación CGI:
  - Recoge los datos, accede a la BD y, con el resultado, produce un documento HTML.
  - Devuelve al servidor el documento precedido de una cabecera HTTP (nph o simple), a través de la salida estándar.
- El servidor Web:
  - Devuelve el resultado, casi directamente, al navegador.

## 2. CGI

## Funcionamiento básico (III)

[illegible]

## Estructura General:

- **Elementos** de entrada de datos.
- **Botón de envío** (Submit).
- **Método de envío** de datos.
- **Acción** que el servidor debe emprender cuando reciba los datos.

# El formulario más simple

Introduzca su nombre:

# Lenguaje de la aplicación CGI

---

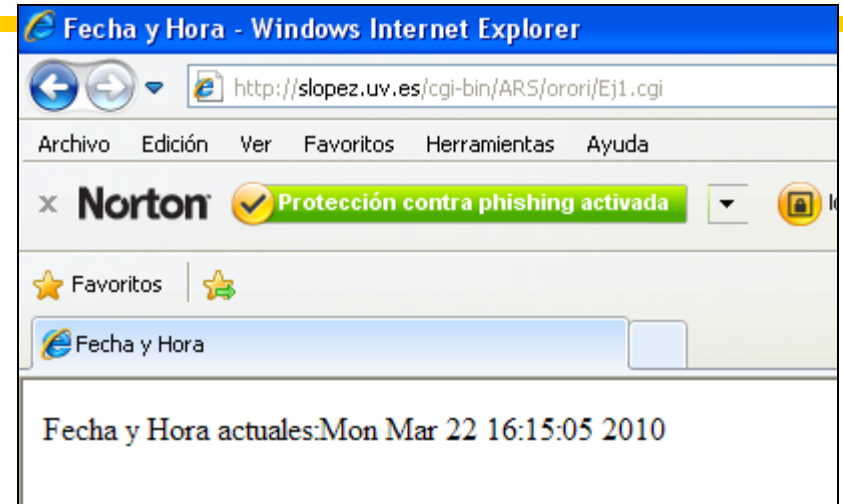
- En principio, no hay restricción. Sólo debe constar:
  - Un modo de leer el contenido de las variables de entorno.
  - Un modo de leer datos por la entrada estándar.
  - Un modo de escribir datos por la salida estándar.
- Compilados:
  - C, C++, Pascal, Fortran, etc.
  - Directamente ejecutables.
  - Son más eficientes
- Interpretados:
  - Shell-Script, Python, Tcl, Awk, etc., pero sobre todo Perl:
    - El Perl es muy potente en el tratamiento de cadenas.
  - La ejecución de la aplicación la realiza el interprete de ordenes (el interprete se pone previamente en funcionamiento). Cuando acaba la ejecución del cgi, también acaba la del interprete.
  - Son más fáciles de depurar, modificar y mantener

## 2. CGI

# Ejemplo de aplicación CGI

```
#include <iostream>
#include <ctime>
using namespace std;
```

```
int main ()
{
    time_t hora;
    time( &hora ); // Almacena la hora y fecha en la
                  //variable
    cout << "Content-Type: text/html" << endl << endl;
    cout << "<html><head>" << endl;
    cout << "<title>Fecha y Hora</title>" << endl;
    cout << "</head><body>" << endl;
    cout << "<p>Fecha y Hora actuales:";
    cout << asctime(localtime(&hora)) << "</p>" << endl;
    cout << "</body></html>" << endl;
    return 0;
}
```



## 2. CGI

# Ejemplo - Variables de Entorno

---

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main() {
printf("Content-type: text/html\n\n");
printf("<html><head><title>Nombre</title></head>");
printf("<body bgcolor=\"#FFFFFF\"><p align=\"center\">");

printf("<br/>SERVER_NAME = %s", getenv("SERVER_NAME"));
printf("<br/>SERVER_SOFTWARE = %s",getenv("SERVER_SOFTWARE"));
printf("<br/>REQUEST_METHOD = %s", getenv("REQUEST_METHOD"));
printf("<br/>QUERY_STRING = %s", getenv("QUERY_STRING"));
printf("<br/>REMOTE_HOST = %s", getenv("REMOTE_HOST"));

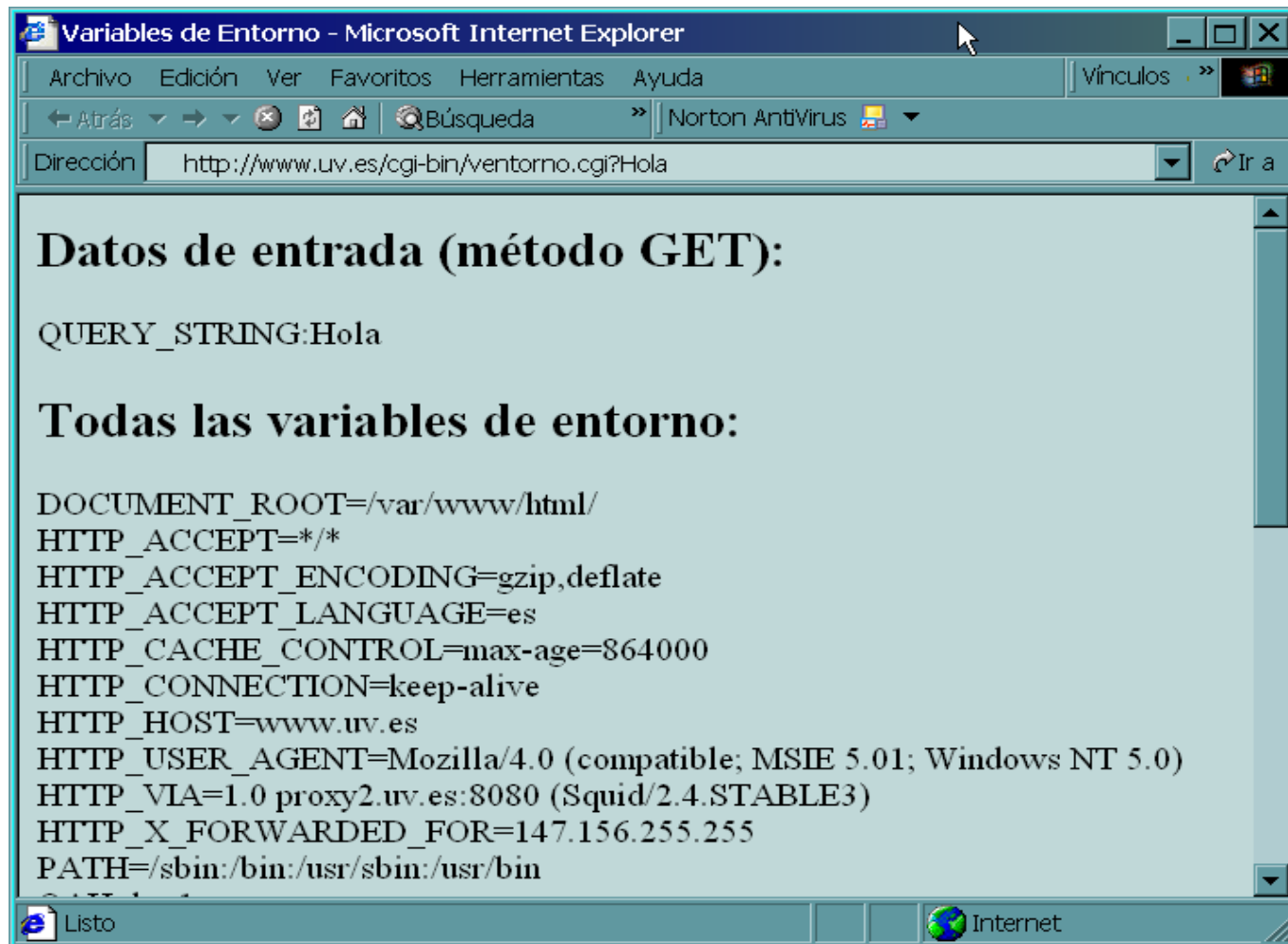
printf("</p></body></html>");
return 0;
}
```

# Imprime las variables de entorno (I)

```
#include <string>
#include <iostream>
#include <cstdlib>
using namespace std;
int main ()
{
    string vEntorno[24] = { "DOCUMENT_ROOT", "HTTP_ACCEPT",
        "HTTP_ACCEPT_ENCODING", "HTTP_ACCEPT_LANGUAGE", ... };
    cout << "Content-Type: text/html" << endl << endl;
    cout << "<html><head><title>Variables de Entorno</title>";
    cout << "</head><body><h2>Datos de entrada(método GET):</h2>";
    cout << "QUERY_STRING:" << getenv("QUERY_STRING") << "<br>";
    cout << "<h2>Todas las variables de entorno</h2>";
    for ( i = 0; i < 24 ; i++ ) {
        cout << vEntorno[i] << ":";
        cout << getenv(vEntorno[i].data()) << "<br>";
    }
    cout << "</body></html>";
}
```

## 6. Ejemplos de aplicaciones CGI

# Imprime las variables de entorno (II)



# QUERY\_STRING

---

## ***Datos de un Formulario:***

*Nombre = Marco*

*Edad = 25*

*Ciudad = Madrid*

*Oficio = funcionario del ayuntamiento*

QUERY\_STRING=

*"Nombre=Marco&Edad=25&Ciudad=Roma&Oficio=funcionario+del+ayuntamiento"*



## 2. CGI

# Ejemplo - Entrada Estándar

---

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main() {
    printf("Content-type: text/html\n\n");
    printf("<html><head><title>Nombre</title></head>");
    printf("<body bgcolor=\"#FFFFFF\"><p align=\"center\">");
    char* metodo, *query;
    //Parámetros en el cuerpo del mensaje (POST)
    //Se identifica el método GET o POST
    metodo = getenv("REQUEST_METHOD");
    //Se obtienen los parámetros del formulario
    if(strcmp(metodo, "GET")==0)
        query = getenv("QUERY_STRING");
    else //METODO POST
    {
        int longitud = atoi(getenv("CONTENT_LENGTH"));
        query = (char*) new char[longitud+1];
        fread(query, 1, longitud, stdin);
    }
    printf(query);
    printf("</p></body></html>");
    return 0;
}
```

# Formas de invocar un CGI

- Directamente desde el Navegador (método GET)
  - Indicando su URL en el campo **Dirección**
- Como destino de un enlace (GET):  
`<a href="/cgi-bin/prueba.cgi"> Enlace al CGI de pruebas </a>`
- Al actuar sobre un mapa activo (mapa de imágenes en el lado del servidor) – (GET):  
`<a href="http://www.uv.es/cgi-bin/mapa_imagenes/campus_burj.cgi">  
 </a>`
- Como dirección origen de una imagen (GET).
  - La salida de la aplicación CGI debe ser de tipo imagen.
  - Es una forma de incrustar **banners**.  
``
- Como acción asociada a un formulario HTML (GET-POST)
  - Es el uso más común de las aplicaciones CGI.
  - El método de invocación (GET o POST) establece la manera en que el CGI recibirá los parámetros.  
`<form action="/cgi-bin/test.cgi" method="post">`

# Resultado de la aplicación CGI

---

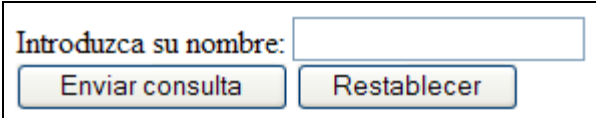
- El CGI devuelve al servidor su resultado a través de la salida estándar
  - C++: **cout**
- El resultado consta normalmente de:
  - Una cabecera HTTP  
Normalmente: **Content-Type: text/html**
  - Una línea en blanco.
  - Un documento MIME completo.
- El servidor completa la información de la cabecera, y el resultado final es enviado al cliente.
- Un CGI puede generar tres tipos de resultados:
  - Un documento MIME completo, con su cabecera completa (nph)
  - Un documento MIME completo, con una cabecera simple.
  - La localización de un documento.

# Ejemplo 3 - CGI (Formulario)

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
<head>
  <title> Ejemplo de cgi </title>
</head>
<body>
  <form action="http://slopez.uv.es/cgi-bin/ARS/orori/ejemplo.cgi" method="post">

    Introduzca su nombre: <input type="text" name="nombre"/> <br/>
    <input name="enviar" type="submit" />
    <input name="borrar" type="reset" />

  </form>
</body>
</html>
```

A screenshot of a web browser displaying the rendered form. It shows the text "Introduzca su nombre:" followed by a text input field. Below the input field are two buttons: "Enviar consulta" and "Restablecer".

# Ejemplo 3 -CGI (C++)

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <iostream>
#include <ctime>
#include <fstream>
using namespace std;
int main()
{
    char* query;
    char* metodo;

    //Se identifica el método GET o POST
    metodo = getenv("REQUEST_METHOD");

    //Se obtienen los parámetros del formulario
    if(strcmp(metodo,"GET")==0)
    {
        query = getenv("QUERY_STRING");
    }
    else //METODO POST
    {
        int longitud = atoi(getenv("CONTENT_LENGTH"));
        query = (char*) new char[longitud+1];
        fread(query,1,longitud,stdin);
    }
}
```

```
string mensaje(query);

//Respuesta del CGI

//Cabecera Simple
cout << "Content-Type: text/html" << endl << endl;

//Página XHTML

cout << "<html><head>" << endl;
cout << "<title> Resultado Ejemplo </title>" << endl;
cout << "</head><body><p>" << endl;
cout << mensaje.c_str() << endl;
cout << "</p></body></html>" << endl;

return 0;

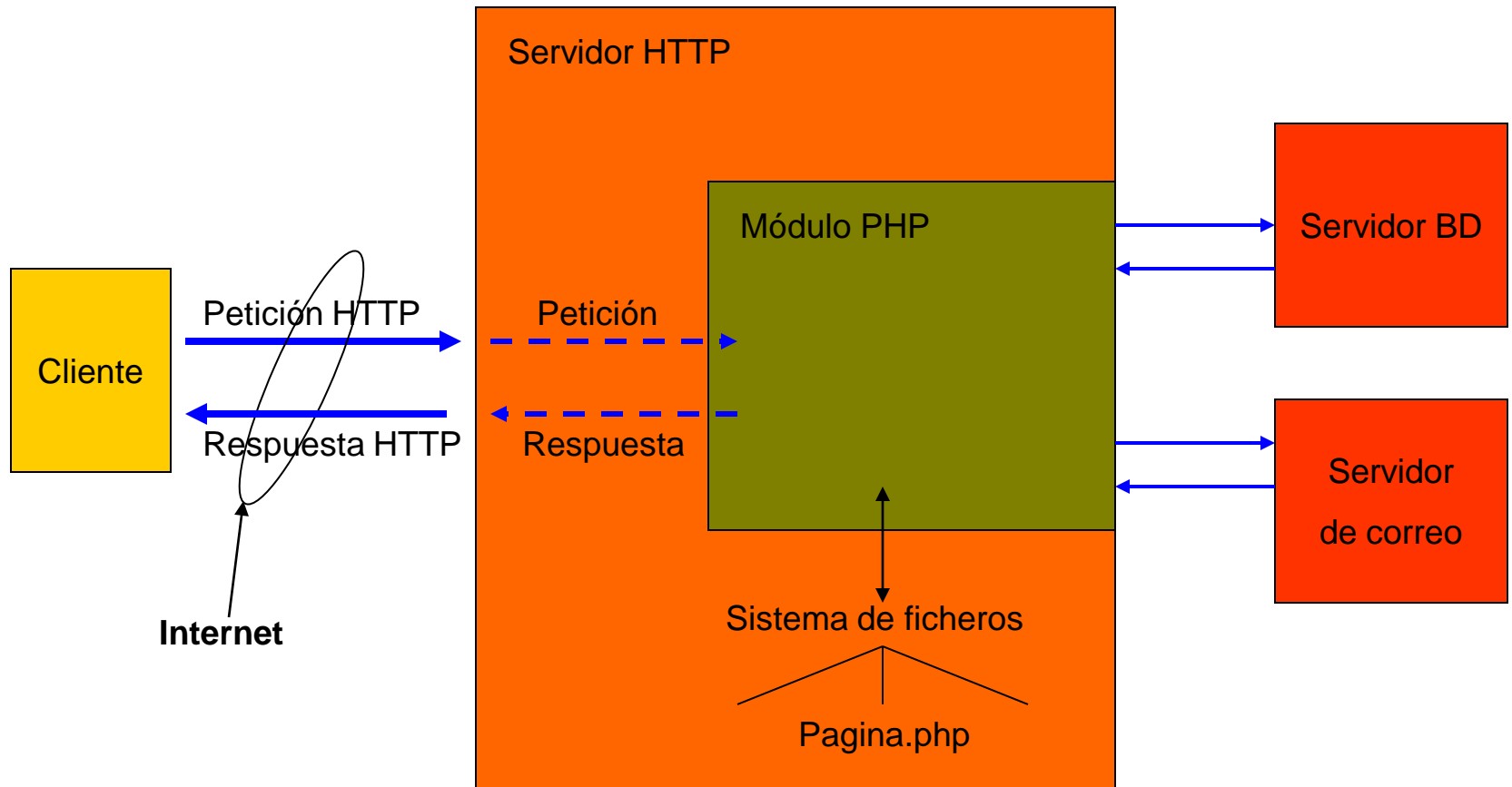
}
```

## Introducción

---

- PHP (Hypertext Preprocessor) fue diseñado para la generación dinámica de contenidos en el lado del servidor.
- A diferencia de los CGIs, con PHP:
  - El contenido dinámico es generado utilizando un lenguaje de script propio (PHP), embebido dentro de código HTML.
  - Las peticiones son procesadas por el mismo servidor
    - El script que genera el código dinámicamente es ejecutado por el propio servidor HTTP.
    - De ello se encarga un módulo incluido en el mismo proceso.
- Permite conectarse a BD y otras aplicaciones externas.
- Primera versión: aparece en la primavera de 1995.
- Última versión: PHP 5.3.2 (4 Marzo 2010)

# Arquitectura PHP



## Ventajas del PHP (I)

---

- PHP es gratis.
  - Licencia de GNU.
- Es de código abierto.
  - El código fuente está disponible, y puede ser modificado libremente.
- Es sencillo.
  - Lenguaje de script con sintaxis similar a Javascript, Perl y C.
  - Diseño modular de fácil ampliación.
- Es embebido.
  - Son páginas HTML que “cambian” al modo PHP cuando es necesario.
- No es un lenguaje de etiquetas
  - A diferencia de ColdFusion.
- Permite no sólo generar HTML sino también imágenes, PDFs, Flash, XML



## Ventajas del PHP (II)

---

- Es estable.
  - Apache/PHP tiene una buena estadística de horas de funcionamiento sin fallos.
  - Las versiones de PHP no cambian radicalmente y de forma incompatible entre versión y versión.
- Es rápido.
  - Es mucho más rápido que la mayoría de scripts para CGI.
  - El intérprete es un módulo dentro del servidor.
- Es multiplataforma.
  - Se puede utilizar con prácticamente cualquier S.O.
  - Integración perfecta con múltiples servidores HTTP.
  - Acceso a múltiples Bases de Datos

## Ventajas del PHP (III)

---

- S.O:
  - Windows, Linux, SunOS, AIX, IRIX, FreeBSD, HP-UX, Solaris, Digital Unix, NetBSD, openBSD, etc.
- Servidores:
  - Apache (UNIX, Win32), ISAPI (IIS, Zeus), NSAPI (iPlanet), SunONE, WebSphere, AOLServer, etc.
- Base de datos:
  - Adabas D, dBase, Empress, IBM DB2, Informix, Ingres, Interbase, Frontbase, mSQL, Direct MS-SQL, MySQL, ODBC, Oracle, PostgreSQL, Velocis, etc.

## Ejemplo

---

```
<html>
<head>
  <title>Ejemplo</title>
</head>
<body>
  <h1> Ejemplo simple.</h1>
  Primer ejemplo de código PHP embebido dentro de
  código HTML.<br/>
  <?php  /* Inicio del código PHP */
  echo "Hola Mundo<br/>"; // comienzo del código
  ....
?>
</body>
</html>
```

## Extensión y localización de los ficheros

---

- El servidor HTTP distingue que el recurso solicitado se trata de una página php, por la extensión.
  - Pasa la petición al módulo PHP para su procesado.
- Extensiones de los ficheros:
  - `.php`
    - Indica código PHP. Extensión genérica.
- Localización de los ficheros:
  - Normalmente, en los mismos directorios donde se ubican el resto de ficheros html.

### 3. PHP: Sintaxis básica

## Delimitadores

---

- El módulo de PHP busca uno de los tags que emplea para reconocer el comienzo de código PHP.
- Ejecuta el código hasta encontrar una marca de final de código
- Continúa por el documento hasta encontrar otra marca de comienzo
- Todo lo que esté fuera de esas marcas queda inalterado

```
<? echo 'delimitador muy común'; ?>
```

```
<?php print("delimitador más recomendable"); ?>
```

```
<script language="php">  
    echo 'otro delimitador';
```

```
</script>
```

```
<% echo 'delimitadores al estilo ASP'; %>
```

- Sólo el segundo y el tercer delimitador están siempre disponibles (los otros dos pueden estar desactivados).

### 3. PHP: Sintaxis básica

## PHP Embebido

---

- El código PHP se puede insertar en cualquier lado del fichero, combinándose con el código HTML de cualquier manera.

```
<?php
    if (date('a') == 'pm') {
        $saludo = 'Buenas tardes/noches!';
    } else {
        $saludo = 'Buenos d&iacute;as!';
    }
?>
<html>
<head><title>Ejemplo</title></head>
<body>
<h1><?php echo $saludo; ?></h1>
</body></html>
```

### 3. PHP: Sintaxis básica

## Primeros elementos

---

- El final de la sentencia está marcada por un punto y coma. La última no lo necesita.

**<?php**

```
print( date("M d, Y H:i:s", time()) );  
print( date("M d, Y H:i:s",  
                                time()  
                                )  
      )  
;
```

**?>**

- Comentarios:

**<?php**

**/\* Comentarios estilo C.**

**Pueden extenderse durante varias líneas.**

**\*/**

**// Comentarios estilo C++. Hasta fin de línea.**

**# Comentarios estilo Perl. Hasta fin de línea...**

## Mayúsculas y minúsculas

---

- Comportamiento mixto en variables y funciones:
  - En las variables, las mayúsculas y minúsculas **IMPORTAN**.

```
$var = 3;  
$Var = 5.6;  
print ("$var $Var");
```

- En los nombres de funciones y palabras reservadas, **NO IMPORTAN**.

```
if ( $a == 4 )  
PRINT ('a vale 4');  
IF ( $a == 3 )  
print ("a vale 3");
```



# Variables (I)

---

- Van precedidas del símbolo \$
- No necesitan ser declaradas.
- No tienen un tipo intrínseco (depende de su valor).
- Las variable utilizadas, antes de ser inicializadas, tienen valores por defecto.

```
$mi_variable = 12 + $var_sin_inicializar; //12
```

- Están débilmente tipadas.

```
$mi_variable = 'Inicializamos como texto';  
$mi_variable = 3; // Entero.  
$mi_variable = 3.14 * $mi_variable; // Real.
```

## 4. PHP: Elementos

# Variables (II)

---

- Ejemplos de conversiones:

```
$var_1 = 123;  
$nom1 = "23Juan";  
$nom2 = "Juan23";  
$var_2 = "12";
```

```
$var_3 = $var_1 * 2 + $var_2;           // 258  
$var_4 = $nom1 . $var_1;                // 23Juan123  
$var_5 = $var_1 + $nom1;                // 146  
$var_6 = $var_1 + $nom2;                // 123
```

# Salida

- **echo**, escribe la cadena que se le pasa como argumento.
  - La cadena puede estar o no incluida entre paréntesis.

```
echo " Mensaje 1 "; echo (" Mensaje 2 ");
```
  - Sin paréntesis, admite varios argumentos.

```
echo "Este mensaje", 'aparecerá', "en el browser";
```
- **print**, similar a echo.
  - Sólo acepta un argumento.

```
print ('3.145159'); print (3.14159);
```
  - El comportamiento de **print** y **echo** es distinto si la cadena va incluida entre comillas simple o dobles:

```
print(" El animal $a tiene $p patas\n"); → Substituye  
print(' El animal $a tiene $p patas\n'); → Literal
```
- **printf**: Salida con formato.

# Operadores (I)

---

- Lógicos:

- **and, or, xor, &&, ||, !**

```
if( $a == 4 and $b && !$c ) ...
```

- Relacionales:

- **==, !=, <, <=, >, >=, ===** (PHP4, identidad), **!==** (no idéntico)

```
$v1 = 10; // Asignación
```

```
$v2 = 10.0;
```

```
$v3 = "10";
```

```
if($v1 == 10) ... // Cierto, son iguales
```

```
if($v2 == 10) ... // Cierto, son iguales
```

```
if($v3 == 10) ... // Cierto, son iguales
```

```
if($v1 === 10) ... // Cierto, son idénticas
```

```
if($v2 === 10) ... // FALSO, el tipo no coincide
```

```
if($v3 === 10) ... // FALSO, el tipo no coincide
```

# Operadores (II)

---

- Aritméticos:

- `++`, `--`, `+`, `-`, `*`, `/`, `%`

```
$a = 3 + 6; ++$a; $a++;
```

- Concatenación de cadenas: `.`

```
$nombre = "Juan" . "Sánchez";
```

- Asignación:

- `=`, `+=`, `-=`, `*=`, `/=`, `%=`, `.=`

```
$v1 += 2; // Equivalente a: $v1 = $v1+2
```

```
$v2 .= 'nuevo'; // Equivalente a: $v2 = $v2 . 'nuevo'
```

# Estructuras de control (I)

---

### ■ Selectivas:

- **if** (condición) {...} **else** {...}
- **if** (...) {...} **elseif** (...) {...} **else** {...}

```
if ( $ciudad == 'Madrid' ) {  
    ...  
} elseif ($ciudad == 'Valencia') {  
    ...  
} else { ... }
```

- **switch** (...) { **case**: .... **break**; .... **default**: .... }
- **FALSE**, 0 , "" o un array vacío, equivalen a un valor de falso. Cualquier otro valor equivale a cierto (**TRUE**).

# Estructuras de control (II)

---

### ■ Bucles:

- **while** (condición) { .. }
- **do** { .. } **while** (condición);
- **for** (.. ; condición ; .. ) { .. }
- **foreach** ( array **as** variable) { .. }

```
$v = array(3, 5, 9, 1);  
foreach ( $v as $elem)  
    print("otro valor: $elem<br/>");
```

# Funciones (I)

---

- En la definición se utiliza la palabra **function**:  
`function nombre_func ( $a1, $a2 ) { ... }`
- Devuelve valores a través de la sentencia **return**
- No hace falta que la función esté definida antes de la línea donde se emplee (PHP 4)
- En el nombre de la función no distingue mayúsculas.
- Se pueden definir dentro de un bloque de condición y entonces no existen hasta que la ejecución pase por él.
- Los parámetros de la función se pueden pasar tanto por valor como por referencia (se marcan con **&**)

```
function Copia (&$destino, $origen) {  
    $destino = $origen;  
    $origen = 0; // No tiene efecto en $b  
}  
$a = 2; $b = 3;  
Copia($a,$b);           // $a = 3, $b = 3
```



## 4. PHP: Elementos

# Funciones (II)

---

- Las variables definidas dentro de la función tiene **ámbito** local. Cualquier variable definida fuera de una función tiene ámbito global.

```
$numero = 5;
function Prueba() {
    $numero = 4;
}
Prueba(); print($numero); //Se escribe 5
```

- Para acceder a las variables globales:
  - Se declaran dentro de la función precedidas de **global**

```
$numero = 5;
function Prueba() {
    global $numero;
    $numero = 4;
}
Prueba(); print($numero); //Se escribe 4
```

- Usamos `static` para evitar que se borre una variable local.

# Arrays en PHP

---

- Como el resto de variables, los array no se declaran, ni siquiera para indicar su tamaño.
- Son dispersos.
  - Los índices de sus elementos no tienen porque ser consecutivos.

```
$vec[1] = '1º elemento';
```

```
$vec[8] = '2º elemento';
```

- Los arrays son asociativos.
  - Los índices no tienen porque ser números

```
$vec['tercero'] = '3º elemento';
```

- Los arrays no son homogéneos.
  - Sus elementos pueden ser de tipos diferentes.

```
$vec[5] = '4º elemento';
```

```
$vec[1000] = 5.0;
```

# Creando arrays

- Hay dos formas de crear un array:
  - Asignación directa.
    - Se añaden sus elementos uno a uno, indicando el índice.

```
$vec[0] = '1º elemento'; $vec[1] = "2º elemento";  
$vec[] = '3º elemento'; //$vec[2]= "3º elem.."
```
  - Utilizando el constructor **array()**.
    - Se añaden entre paréntesis los primeros elementos del array. El primero de todos tiene asignado el índice cero.

```
$vec = array ( 3, 9, 2.4, 'Juan');  
// $vec[0] = 3; $vec[1] = 9; $vec[2] = 2.4;...
```
    - Se pueden fijar el índice con el operador **=>**

```
$vec = array ( 2=>3 ,9, 2.4, 'nombre'=>'Juan');  
// $vec[2] = 3;$vec[3]=9;.. $vec['nombre']='Juan'
```

Ejemplo1: `$unarray = array("dia" => 15, 1 => "uno");`

Ejemplo2: `$otro = array("unarray" => array(0=>14, 4=>15), "nombre" => "Una tabla");`

# Modificando arrays

---

Para eliminar un elemento del array hay que emplear unset()

- `unset($miarray ['nombre']);`
- `unset($miarray );`

Existen funciones para añadir, borrar elementos al inicio y final de un array.

Añadir Elemento:

- al final : `array_push($miarray, valor);`
- al inicio: `array_unshift($miarray, valor);`

Para eliminar Elemento (también lo devuelve):

- del final: `array_pop($myarray);`
- del inicio: `array_shift($miarray);`

# Recorriendo arrays

### ■ Secuenciales:

- La función **count()** determina el número de elementos del array.
- Se recorren utilizando un bucle cualquiera, comenzando por el primer elemento (índice 0).

```
$v = array ( 3 ,9, 2.4, 'Juan' );  
for ($i = 0 ; $i < count($v); $i ++ )  
    print("<br/>v[$i]=$v[$i]");
```

### ■ No secuenciales:

- Para recorrer todos sus elementos en orden se puede utilizar: **foreach** ( array **as** índice=>valor)

```
$v = array ( 8=>3 ,9, 1=>2.4, 'nombre'=> 'Juan' );  
foreach ($v as $i=>$valor)  
    print("<br/>v[$i]=$valor");
```

```
foreach ($v as $valor)  
    print("<br/>$valor");
```

# Arrays predefinidos (I)

---

- Array predefinidos dentro de cualquier aplicación PHP:
  - **\$HTTP\_GET\_VARS** o **\$\_GET** (PHP 4.1): contiene los valores enviados por el método GET.  

```
print('Variables enviadas por el método GET');  
foreach($HTTP_GET_VARS as $nom_variable=>$valor)  
{  
    echo "$nombre_variable = $valor<br>\n";  
}
```
  - **\$HTTP\_POST\_VARS** o **\$\_POST** (PHP 4.1): contiene los valores enviados por el método POST.
  - **\$HTTP\_COOKIE\_VARS** o **\$\_COOKIE** (PHP 4.1): contiene los valores del cookie.
  - **\$\_REQUEST** (PHP 4.1): Contienen todas las variables incluidas en los tres anteriores. (*\$\_GET*, *\$\_POST*, y *\$\_COOKIE*)

# Arrays predefinidos (II)

---

- **\$HTTP\_SERVER\_VARS** o **\$\_SERVER** (PHP 4.1): variable asociadas a la cabecera HTTP o a variables del servidor.  
  
`$acepta = $HTTP_SERVER_VARS ['HTTP_ACCEPT'];`  
`$cliente = $_SERVER['HTTP_USER_AGENT'];`  
`$camino_raiz = $_SERVER['DOCUMENT_ROOT'];`  
`$fichero_php = $_SERVER['PHP_SELF']; //SCRIPT_NAME`
- **\$HTTP\_SESSION\_VARS** o **\$\_SESSION**(PHP 4.1): variables de la sesión.
- **\$HTTP\_ENV\_VARS** o **\$\_ENV** (PHP 4.1): variables de entorno.  
`$camino = $_ENV['PATH'];`
- **\$GLOBALS**: contiene todas las anteriores, además del resto de variables globales.

# Formularios (I)

---

- Las variables del formulario son transformadas por el servidor en elementos de los arrays `$_GET` o `$_POST`, asignándole como índices el mismo nombre que las variables del formulario.

```
<!-- formulario.html -->
<form action="accion.php" method="GET">
Su nombre: <input type="text" name="nombre"/><br/>
Su edad: <input type="text" name="edad"/><br/>
<input type="submit"/>
</form>
```

```
-----
<!-- accion.php -->
.....
<?php
    echo"<p>Hola ", $_GET['nombre'], "</p>";
?>
<p>Tienes <?php print($_GET['edad']);?> años</p>
```



# Formularios (II)

- Los valores de las listas de selección se guardan en un array.

```
<form action="accion.php" method="POST">
<select multiple name="menu[]">
<option>Tortilla </option>
<option>Paella </option>
<option>Fabada </option>
<option>Lentejas </option>
</select><input type="submit"/></form>
```

```
-----
<?php
    echo 'Platos seleccionados:<br/>';
    foreach( $_POST['menu'] as $plato )
    {
        echo "$plato<br/>\n";
    }
?>
```

# Formularios (III)

---

Ejemplos equivalentes empleando `$_REQUEST`, válidos para el método de envío GET y POST

-----  
`<!-- accion.php -->`

`<?php`

`echo"<p>Hola ", $_REQUEST['nombre'], "</p>";`

`?>`

`<p>Tienes <?php print($_REQUEST['edad']);?> años</p>`

-----  
`<!-- accion.php -->`

`<?php`

`echo 'Platos seleccionados:<br/>';`

`foreach( $_REQUEST['menu'] as $plato )`

`{`

`echo "$plato<br/>\n";`

`}`

`?>`

# Función isset

---

La función `isset` determina cuando una variable ha sido definida previamente, y además, tiene un valor diferente de `NULL`.

```
$var1 = "hola";

if (isset($var1))          //TRUE
...

if (isset($var1,$var2)) //FALSE
...

$var1 = NULL;

if (isset($var1))          //FALSE
...
```

## 4. PHP: Elementos

# Ejemplo

```
<html> <head> <title> Saludo </title> </head> <body>
<?php
    if ( !isset(['nombre']) ) {
?>
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="POST">
Introduce tu nombre:<input type="text" name="nombre"/>
<input type="submit"/></form>
<?php
    } // end_if ( !$_POST['nombre'] )
    else {
        $usuario = $_POST['nombre'];
?>
Bienvenido
    <?php echo $usuario; ?>
        } // end_else
?>
<br/>
</body></html>
```

## 4. PHP: Elementos

# Envío de email

---

Con php se puede realizar el envío de un email con la función mail:

```
mail(destinatario/s, asunto, texto del mensaje[, cabeceras adicionales [, parámetros adicionales]]);
```

Los parámetros de la función serán:

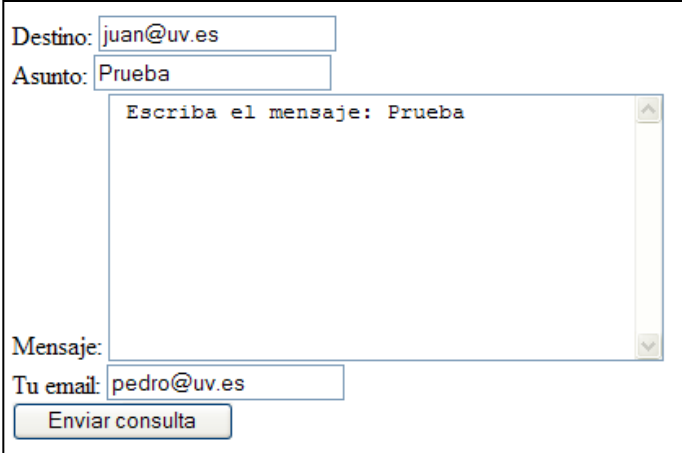
- Destinatario/s: La dirección email del destinatario o destinatarios.
- Asunto: El asunto del mensaje.
- Texto del mensaje: mensaje en texto plano o xhtml.
- Cabeceras adicionales (opcional) : From, CC, Reply-To, ... (separados por "\r\n").
- Otros parámetros adicionales. (opcional).

```
<?php  
    mail("juan@hotmail.com", "Saludos", "Hola como va todo ...", "From: pepe@hotmail.com\r\n");  
?>
```

## 4. PHP: Elementos

# Envío de email - Ejemplo (I)

```
<html>
  <head>
  </head>
  <body>
    <form action="email.php" method="GET">
      Destino: <input type="text" name="destino"/><br/>
      Asunto: <input type="text" name="asunto"/><br/>
      Mensaje: <textarea name="contenido" rows="10" cols="40"/> Escriba el
mensaje: </textarea> <br/>
      Tu email: <input type="text" name="origen"/><br/>
      <input type="submit"/>
    </form>
  </body>
</html>
```



The screenshot shows a web browser window displaying the HTML form. The 'Destino' field contains 'juan@uv.es', the 'Asunto' field contains 'Prueba', and the 'Mensaje' field contains 'Escriba el mensaje: Prueba'. The 'Tu email' field contains 'pedro@uv.es'. A button labeled 'Enviar consulta' is at the bottom.

## 4. PHP: Elementos

# Envío de email - Ejemplo (I)

---

```
<html><head></head> <body>
<?php
if (isset($_REQUEST['destino'], $_REQUEST['asunto'], $_REQUEST['contenido'], $_REQUEST['origen']))
{
    $destino = $_REQUEST['destino'] ;
    $asunto = $_REQUEST['asunto'];
    $contenido = $_REQUEST['contenido'];
    $origen = $_REQUEST['origen'];
    $cabeceras = "From: $origen\r\n";
    $cabeceras .= "Content-type: text/html;charset=iso-8859-1\r\n";
    $cabeceras .= "Reply-To: $origen\r\n";

    mail($destino,$asunto,$contenido,$cabeceras);
    echo "mensaje enviado correctamente <br/>";
}
else
    echo "Error al enviar el mensaje";
?>
</body></html>
```

# Inclusión de elementos

---

Es posible agrupar código php en ficheros e insertarlos en otras páginas php, de forma que se pueda reutilizar el código sin necesidad de escribirlo de nuevo en las páginas php.

Para realizar la inserción se utiliza la función include.

Esta función se usa habitualmente para dar uniformidad a las páginas php que forman parte de un espacio web, proporcionando una misma estructura (índice, cabecera, pie de página, ...) a todas ellas.

....

```
<?php include("menu.txt"); ?>
```

```
<?php include ("librería.php"); ?>
```

....

El fichero se incluye en su totalidad en la posición del include, como si se hubiera realizado una copia de su contenido.



# Acceso Autorizado

---

Es posible restringir el acceso autorizado a las páginas php empleando el sistema de autenticación de HTTP.

Para ello se hará uso de las variables PHP\_AUTH\_USER, PHP\_AUTH\_PW, que se encuentran en el vector \$\_SERVER.

```
<html><head></head><body>
<?php
    if (!isset($_SERVER['PHP_AUTH_USER']))
    {
        header('HTTP/1.0 401 Unauthorized');
        header('WWW-Authenticate: Basic realm="Acceso restringido"');
        echo 'Authorization Required.';
    }
```

## 4. PHP: Elementos

# Acceso Autorizado

---

```
else
{
    if (($_SERVER['PHP_AUTH_USER']!="ars") || ($_SERVER['PHP_AUTH_PW']!="ars"))
    {
        header('WWW-Authenticate: Basic realm="Acceso restringido");
        header('HTTP/1.0 401 Unauthorized');
        echo "Acceso Incorrecto<br/>";
    }
    else
    {
        echo "Acceso correcto<br/>";
        echo "Ha introducido el nombre de usuario: ";
        echo $_SERVER['PHP_AUTH_USER'];
        echo "<br/> Ha introducido la contraseña: ";
        echo $_SERVER['PHP_AUTH_PW'];
    }
}
?>
</body></html>
```

## 4. PHP: Elementos

# Sesiones

---

Las sesiones permiten almacenar la información relacionada con las actividades llevadas a cabo por un usuario en un servidor web mientras está conectado al mismo.

Una vez se cierra el navegador, la sesión se destruye.

Para inicializar la sesión se emplea la función:

- explícitamente: `session_start()`.
- al introducir una variable en una sesión con la función:  
`session_register('mi_variable')`.

En ambos casos, si la sesión no existe se crea una nueva. Si no, se utilizará la sesión actual.

Además de la función `session_register`, para introducir nuevas variables en la sesión se puede emplear el vector `$_SESSION`. (php 4)

`$_SESSION['mi_variable'] = mi_valor.`

# Sesiones - Ejemplo

---

```
<html><head><title>Contar p&acute;ginas vistas por un usuario</title> </head><body>
<?php
    session_start();
    if (!isset($_SESSION['contador'])) {
        $_SESSION['contador'] = 1;
    } else {
        $_SESSION['contador']++;
    }
    echo "Has visto " . $_SESSION['contador'] . " p&acute;ginas";
?>
<br/>
<br/>
<a href="contador.php">Ver otra p&acute;gina</a>
</body>
</html>
```

## 4. PHP: Elementos

# Cookies

En php podemos utilizar cookies a través de la función `setcookie()`.

`setcookie( string nombre [, string valor [, int expire = 0 [, string path [, string dominio [, bool secure=false [,bool httponly = false ]]]]])`

- nombre: nombre de la cookie.
- valor: valor asociado a la cookie.
- expire: tiempo de expiración de la cookie medido en segundos UNIX (Generalmente se utiliza la función `time()` + segundos de validez).
- path: camino donde tiene valor la cookie.
- dominio: dominio donde tiene valor la cookie.
- secure: sólo se transmitirá la cookie sobre canales seguros (https).
- httponly: si es true sólo se permitirá el acceso a la cookie cuando se utilice el protocolo http. Es decir, para lenguajes de script como javascript, la cookie no estaría accesible.

## 4. PHP: Elementos

# Cookies

---

```
<html><head><title>Contar p&acute;ginas vistas por un usuario</title></head><body>
<?php
    if (!isset($_COOKIE['contador'])) {
        echo "Has visto 1 p&acute;gina";
        setcookie("contador",2,time()+3600);
    } else {
        echo "Has visto " . $_COOKIE['contador'] . " p&acute;ginas";
        setcookie("contador",$_COOKIE['contador']+1,time()+3600);
    }
<br/> <br/>
<a href="cookie.php">Ver otra p&acute;gina</a>
</body>
</html>
```

## 4. PHP: Elementos

# Ficheros

---

En php existen las funciones habituales para tratamientos de ficheros:

`bool file_exists (string $nombre_fichero)` : indica si un fichero existe.

`manejador fopen (string $nombre_fichero, string modo)` : abre un fichero.

los modos que se pueden emplear son: r, r+, w, w+, a, a+, x, x+.

`bool fclose (manejador $fichero)`: cierra un fichero.

`string fgets(manejador $fichero [, int $longitud])` : lee del fichero hasta llegar a final de línea o a longitud especificada.

`string fputs(manejador $fichero, string $cadena [, int $longitud])`: escribe en el fichero toda la cadena, o la longitud especificada.

`bool feof( manejador $fichero)`: indica cuando se ha llegado al final del fichero.

## 4. PHP: Elementos

# Ficheros - Ejemplo

---

//Lectura y Escritura de Fichero (copia de un fichero a otro)

```
if (file_exists("datosIn.txt"))
{
    $ficheroIn = fopen("datosIn.txt","r");
    $ficheroOut = fopen("datosOut.txt","w");

    while (!feof($ficheroIn))
    {
        $linea = fgets($ficheroIn);
        fputs($ficheroOut, $linea);
    }
    fclose($ficheroIn);
    fclose($ficheroOut);
}
```



# Más información

---

- Más información sobre CGIs:
  - CGI Programming on the World Wide Web.  
Shishir Gundavaram. O'Reilly.  
<http://www.oreilly.com/openbook/cgi/>
  - CGI Programming 101.  
<http://www.cgi101.com/class/>
- Más información sobre PHP:
  - PHP Bible  
Tim Converse and Joyce Park. Hungry Minds Inc.
  - Manual de PHP  
<http://www.php.net/docs.php>